EXTENDING PROPAGATION WITH USER-DEFINED EQUATIONS, APPLICATIONS TO OPTIMIZATION AND PARTIAL DERIVATIVES COMPUTATION

Véronique Pommier-Maurussane (1), Luc Maisonobe (2), Pascal Parraud (3)
(1)(2)(3) CS Communication&Systems, Parc de la Plaine, 5 rue Brindejoncs des Moulinais, BP 15872, 31506 Toulouse cedex 5, Veronique.Pommier@c-s.fr , Luc.Maisonobe@c-s.fr, Pascal.Parraud@c-s.fr

Abstract: *This paper presents a new way to deal with transition matrices handling in variational equations. While working simultaneously on several problems dealing with orbit propagation: low-thrust trajectories and orbit determination, we implemented a feature allowing to add user equations to a propagator in order to solve the first problem. Then it appeared that this feature could be reused to deal with the second one. Indeed, the orbit determination problem is based on variational equations involving transition matrices, which can also be considered as additional parameters, propagated at the same time as the original state vector. This method allows a very modular implementation of both problems, with looser coupling in the equations. It has been successfully integrated in the Orekit open-source library.*

Keywords: *Jacobians, propagation, optimization, customization, Orekit.*

1 Introducing the problem

High accuracy orbit propagation often involves solving Initial Value Problems (IVP) using the motion equations with various force models. However this is not sufficient to solve some specific problems like low thrust trajectory optimization with boundary constraints which involve both solving an optimal control problem (the optimization part) and solving a Two Point Boundary Value Problems (TPBVP, the boundary constraints part). This is also not sufficient for orbit determination which involves computing the derivatives of the orbital state vector throughout the propagation time with respect to initial state and to models parameters.

For the first problem type, the optimal control problem, in addition to position-velocity parameters, extra parameters have to be integrated: the dual parameters of the Pontryagin principle. These parameters have their own differential equations that should be added to the classical equations of motion.

For the second problem type, the Two Points Boundary Value Problem, initial parameters need to be adjusted in order to have the final state vector reach the desired boundary condition. This is done by solving a non-linear optimization problem. The best algorithms dealing with such problems require computation of a Jacobian matrix (from worst to best algorithm: steepest descent, conjugate gradient, Gauss-Newton, and Levenberg-Marquardt). This matrix represents the final state derivative with respect to the initial state.

For the third problem type, the orbit determination problem, initial parameters and force models parameters need to be adjusted as well, in order to have small residuals for all measurements. This is done by least squares problems solving or Kalman filtering. These algorithms also need Jacobian matrices. They use the current state derivative with respect to the initial date and the current state derivative with respect to the force models parameters at each measurement time.

In the last two cases, Jacobians are computed either only at final states or during all propagation. These matrices are sometimes called *transition matrices:* $\Psi_y(t)$ *and* $\Psi_p(t)$. They make the connection between the propagated state and both initial state and force models parameters. They are time-dependent matrices following orbit propagation: $dy(t)/dy_0$ et $dy(t)/dp$, where y stands for the state vector, y_0 this state vector initial value at t_0 and p a parameters vector (for example drag coefficients).

2 Straightforward implementation

The spacecraft trajectory is computed by integrating the Ordinary Differential Equations (ODE) defined by:

$$\frac{dy(t)}{dt} = f(t,y) \Rightarrow y(t) = \int f(\tau, y)\, d\tau$$

The common way to solve the TPBVP is to compute the Jacobian of the final state $\Psi_y(t)$ and $\Psi_p(t)$ and use them in an optimization algorithm to minimize $\|y(t) - \hat{y}(t)\|$ where $\hat{y}(t)$ is the expected final state.

The final state Jacobians are defined as:

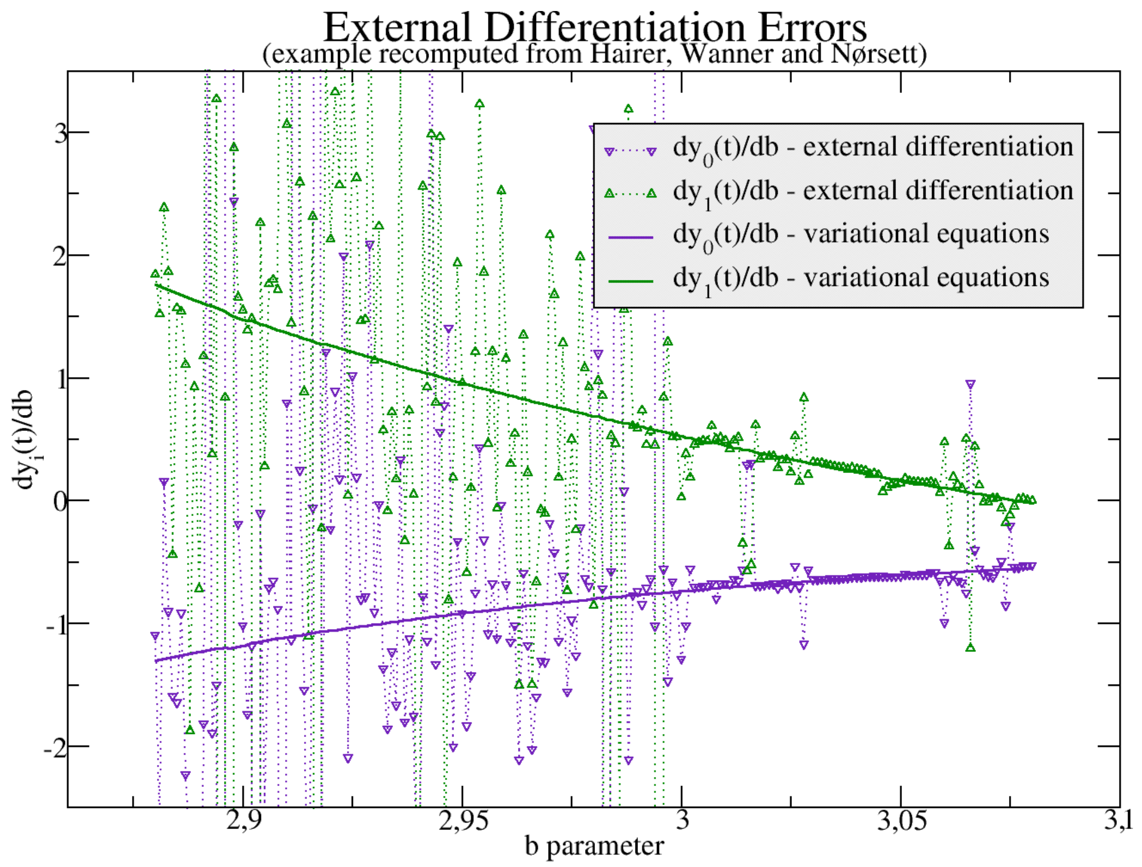
$$\psi_y(t) = \frac{\partial y(t)}{\partial y_0(t)}$$
$$\psi_p(t) = \frac{\partial y(t)}{\partial p}$$

The first way to compute these matrices is by finite differences. A first central trajectory y(t) is computed by a simple solver from an initial state $y_0(t)$, then this initial state is slightly shifted to get several close trajectories. The final states of all those trajectories are combined together to compute the Jacobian matrix $\Psi_y(t)$ and $\Psi_p(t)$ by finite differences. In the Ordinary Differential Equations (ODE) world, this is known as *external differentiation*.

It is well-known since more than 20 years (Hairer, Wanner and Nørsett 1987) that numerical stability of external differentiation methods is very poor when using modern adaptive step size integration methods. This is due to the noise introduced when different initial states lead to different conditional branches used in the step size control.

The following figure[1] shows this behavior. The smooth lines are the exact derivatives and the noisy curves are the derivatives computed by external differentiation.

1 This figure is inspired by Hairer, Wanner and Nørsett own example



The common way to solve the orbit determination problem is to compute the differential equations that govern the evolution of the Jacobian matrices at the same time as the main problem is solved, thus preserving consistency. These equations are called *variational equations* $d\Psi_y(t)/dt$ *and* $d\Psi_p(t)/dt$ in which $\Psi_y(t)$ and $\Psi_p(t)$ are the transition matrices defined previously. The variational equations are defined as follows:

$$\frac{dy(t)}{dt}=f(t,y) \quad y(t)=\int f(\tau,y)d\tau$$

$$\psi_y=\frac{\partial y(t)}{\partial y_0} \qquad \frac{d\psi_y}{dt}=\frac{\partial f}{\partial y}\psi_y$$

$$\psi_p=\frac{\partial y(t)}{\partial p} \qquad \frac{d\psi_p}{dt}=\frac{\partial f}{\partial y}\psi_p+\frac{\partial f}{\partial p}$$

where $\Psi_y(t)$ is the matrix $dy(t)/dy_0$ and $\Psi_p(t)$ the matrix $dy(t)/dp$.

- $\frac{\partial f}{\partial y}=J_y$ is the partial derivatives of the time derivative f with respect to the state y;
- $\frac{\partial f}{\partial p}=J_p$ is the partial derivatives of the time derivative f with respect to the force models parameters p.

These equations show that the state global Jacobian time derivatives is linked to the local Jacobian of the state time derivatives.

The Jacobian matrices $J_y(t)$ and $J_p(t)$ can be computed gradually throughout the propagation.

The transition matrices initial values are the identity matrix for dy(t)/dy0 at t_0 and the null matrix for dy(t)/dp at t_0. From these initial values, the variational equations compute the values at any time. However, if one considers that the user propagates section by section from t_0 to t_1, then from t_1 to t_2 and so on, he may want to have all his matrices computed with respect to the initial time t_0.

This is typically what happens for orbit restitution: the initial time corresponds to the expected orbit adjustment time. It is thus necessary to have an option for setting the transition matrices to any initial value. In the case defined previously, the user will start each section by setting the matrices to the current value of the matrices obtained at the end of previous section.

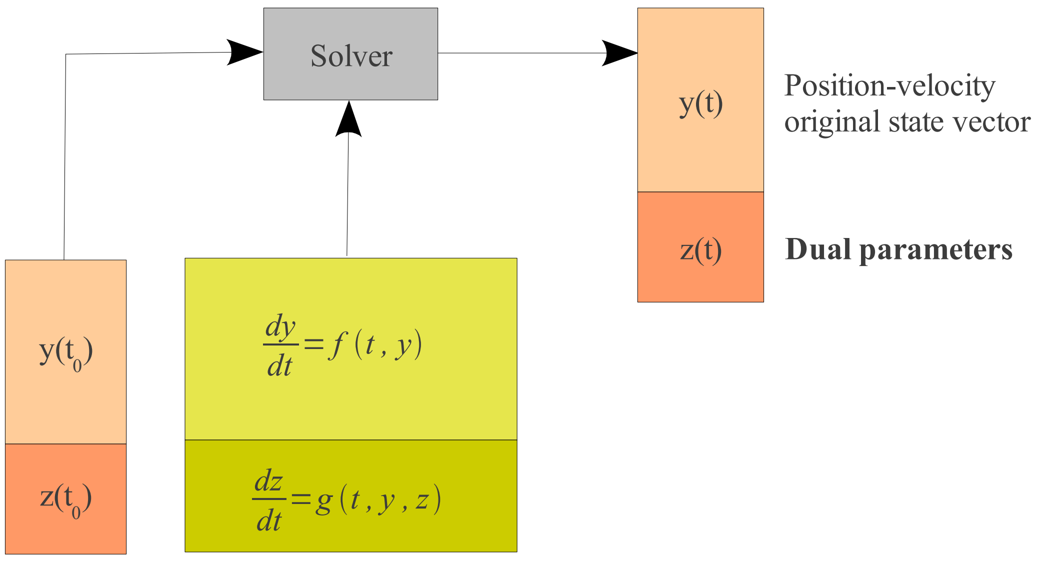
When these Jacobian matrices cannot be computed explicitly, finite differences are used, which is then called internal differentiation.

This method has been implemented almost everywhere, it works well but has some drawbacks and is clearly not extensible. One of the problems is that the differentiation process is fully embedded in the core propagation equations. Switching from one force model to another or changing the ODE solver thus implies numerous adaptations and validations. This is even more difficult for the optimal control problem as the equations for dual parameters are complex, problem-dependent and can almost never be differentiated analytically. That's why common propagation solvers cannot be used for low thrust trajectories with boundary constraints, and very specific tools are usually developed.

3 A bypass resolution

3.1 Additional equations handling

The low-thrust trajectory problem can be solved by adding a set of extra parameters, the dual parameters, to the state vector. Via the corresponding differential equations, these parameters are then included in the propagation process. To allow that feature, it is only necessary to make a few changes in the ODE solver. These changes include handling an array of equations and extending the state vector to take these parameters into account. This method is quite easy to implement and to validate.



As shown in the previous diagram, the additional equations for additional parameters evolution

$\frac{dz}{dt}=g(t,y,z)$ can depend both on the original state vector y(t) and on the additional parameters z(t). On the contrary, the original equations for state vector evolution $\frac{dy}{dt}=f(t,y)$ only depend on the state vector y(t) itself and have no relations with the additional parameters.

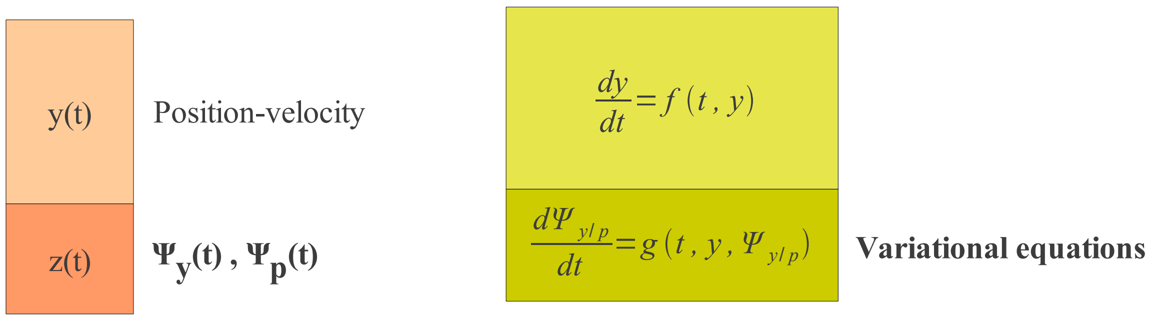
In the orbit determination problem case

Adding equations is not merely extending the size of an array and providing one function to compute its derivatives. There is a fly in the ointment as far as adaptive step size is concerned. Adaptive step size is done by estimating a local error on the state vector. If the error exceeds a predefined threshold, the step is rejected, a smaller step size is computed and used for another attempt on the current step. As adding equations extends the state vector size, the extra parameters are included in the error estimation. It is often very difficult to specify a threshold for these parameters, as they have almost no physical meaning. From a propagation standpoint, there is also no real need for including these parameters into error estimation, which should be based on original position-velocity state only.

Some modern solvers provide continuous output models between steps using dedicated interpolators. In that case, the additional equations must be provided to the integrator in order to propagate the whole state vector. This allows for example to compute residuals in orbit determination, we will see how in next section.

3.2 Application to the transition matrices problem

While working on both variational equations and low-thrust trajectories problem, it appeared that somehow, they were similar problems and could be solved using the same mechanism. Basically, they fit in a single frame which consists in adding parameters to the state vector, and adding to the ODE solver the associated differential equations to propagate them. In that case, the additional parameters are the elements of the $\Psi_y(t)$ and $\Psi_p(t)$ matrices, and the corresponding equations are the variational equations defined previously :



That mechanism is very interesting for orbit determination, when applied to residuals estimation and model parameters estimation. Orbit determination is an iterative process, each iteration being a propagation initialized from the current estimated orbit. During propagation, at each integration step the local model from the ODE solver is used to compute both current state and partial derivatives at the current measurement time. These are used to update the normal equations, which are used by the upper optimizer to adjust the orbit estimation. Computing the partial derivatives $\Psi_y(t)$ by additional equations mechanism allow looser coupling between the core propagator and the other components of the orbit determination system. Adding estimated model parameters is also very simple due to the modular structure of the equations handling.

During orbit propagation, the evolution of the state vector is given by differential equations coming from force models. Variational equations need local Jacobians of the state vector time derivatives (J_y and J_p matrices), which are computed from the force models Jacobians. If analytical equations are available, these Jacobians can be computed directly, otherwise they can still be computed by finite differences, for the original state vector as well as for additional parameters. Corresponding steps can either be user-specified or computed automatically. During a single propagation, both cases can occur, as some force models are more complex than others.

All the matrices involved are computed with respect to Cartesian parameters, even if propagation is done in equinoctial parameters. In that case, a post-processing conversion is required.

4 Conclusion

We have shown that low thrust trajectory problem could be handled by a classical propagator using a simple feature: the additional equations, which can also be applied to the partial derivatives computation in the orbit determination problem.

This method has been fully implemented in version 5.1 of the Orekit open-source library which release is scheduled in early 2011.

5 References

[1] Véronique Pommier-Maurussane and Luc Maisonobe, “ Orekit : an Open-source Library for Operational Flight Dynamics Applications ”, Proceedings 4th International Conference on Astrodynamics Tools and Techniques - 4th ICATT, Madrid, Spain, 2010.

[2] David A. Vallado, “Fundamentals of Astrodynamics and Applications”, Space Technology Library, 2007

[3] Hairer, Wanner and Nørsett, “ Solving Ordinary Differential Equations I – Non-Stiff Problems”, Springer Verlag, 1987